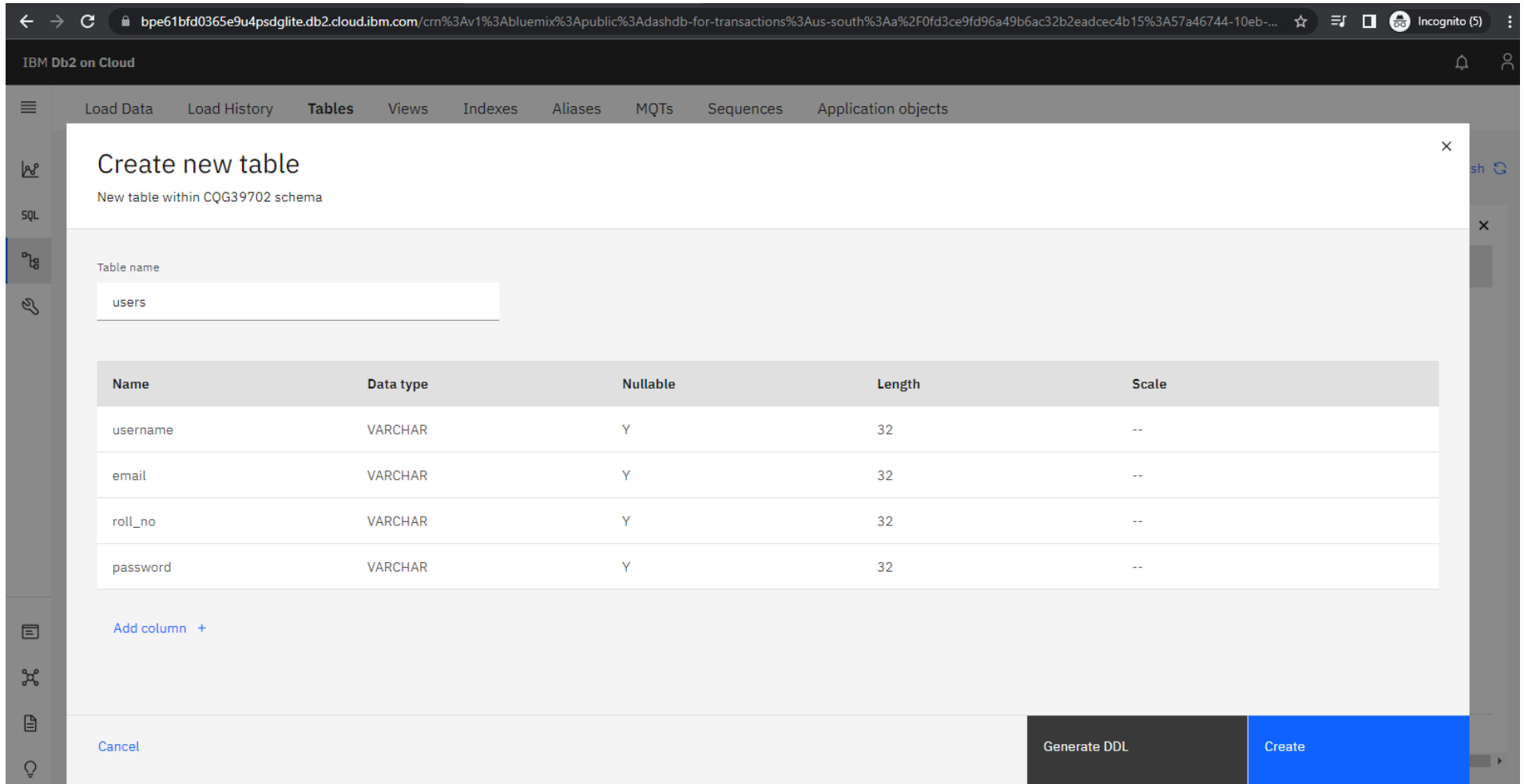


## Assignment – 2

DATE	30 <sup>th</sup> September, 2022
TEAM ID	PNT2022TMID35555
PROJECT NAME	Inventory Management System for Retailers
MAXIMUM MARKS	2 Marks

## Question 1

Create a table in IBM cloud with username, email, password, roll number



IBM Db2 on Cloud

Load Data Load History **Tables** Views Indexes Aliases MQTs Sequences Application objects

### Create new table

New table within CQG39702 schema

Table name

users

Name	Data type	Nullable	Length	Scale
username	VARCHAR	Y	32	--
email	VARCHAR	Y	32	--
roll_no	VARCHAR	Y	32	--
password	VARCHAR	Y	32	--

[Add column +](#)

[Cancel](#) [Generate DDL](#) [Create](#)

Load DataLoad History**Tables**ViewsIndexesAliasesMQTsSequencesApplication objects

Find schemas or tables

Refresh

Schemas

<input checked="" type="checkbox"/>	Name	Type	Tables ▲
<input checked="" type="checkbox"/>	CQG39702	User	1

Total: 1, selected: 1

Tables

New table

<input type="checkbox"/>	Name ▼	Schema	Properties
<input type="checkbox"/>	USERS	CQG39702	...

Total: 1, selected: 0

## Populating the table

The screenshot displays the IBM Db2 on Cloud web interface. The browser address bar shows the URL: `bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aus-south%3Aa%2F0fd3ce9fd96a49b6ac32b2eadcec4b15%3A57a46744-10eb-...`. The page title is "IBM Db2 on Cloud".

The interface is divided into several sections:

- Left Sidebar:** Contains navigation icons for SQL, Tables, Views, MQTs, Aliases, and Nicknames. The "SQL" icon is currently selected.
- Data objects panel:** Shows a search bar "Filter objects" and a list of objects under the "Data objects" tab. The list includes "CQG39702", "Tables", "USERS", "Views", "MQTs", "Aliases", and "Nicknames".
- Code Editor:** A central area with a tab labeled "\*Untitled - 1". It contains the following SQL code:

```
1 INSERT INTO users
2 VALUES ('hari@gmail.com', 'hari', '001', 'erhf');
3
4 INSERT INTO users
5 VALUES ('prem@gmail.com', 'prem', '002', 'erhffg');
6
7 INSERT INTO users
8 VALUES ('arvind@gmail.com', 'arvind', '003', 'wefw');
9
10 INSERT INTO users
11 VALUES ('nehanth@gmail.com', 'nehanth', '004', 'wefwe');
```

The code is syntax-highlighted, and the last line is selected. A "Syntax assistant" toggle is visible in the top right of the editor area.
- Run Button:** A blue button labeled "Run all" is located in the top right corner of the code editor.
- History Panel:** A small panel at the bottom of the code editor area is labeled "History".

☰

[📊](#)

[SQL](#)

[🔧](#)

[🔗](#)

[📄](#)

[💡](#)

Load Data

Load History

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

CQG39702.USERS [Back](#)

[🗑️](#)

[Export to CSV](#) [⬇️](#)

USERNAME	EMAIL	ROLL_NO	PASSWORD
arvind@gmail.com	arvind	003	wefw
hari@gmail.com	hari	001	erhf
nehanth@gmail.com	nehanth	004	wefwe
prem@gmail.com	prem	002	erhffg

## Question 2

### Perform Update and Delete Queries

The screenshot shows the IBM Db2 on Cloud web interface. The browser address bar displays the URL: `bpe61bfd0365e9u4psdglite.db2.cloud.ibm.com/crn%3Av1%3Abluemix%3Apublic%3Adashdb-for-transactions%3Aus-south%3Aa%2F0fd3ce9fd96a49b6ac32b2eadcec4b15%3A57a46744-10eb-...`. The interface is titled "IBM Db2 on Cloud".

On the left sidebar, the "SQL" tab is selected. Under "Data objects", the database "CQG39702" is listed, containing "Tables", "USERS", "Views", "MQTs", "Aliases", and "Nicknames".

The main editor area shows a query in a file named "\*Untitled - 1". The query is:

```
1 UPDATE users
2 SET password='wenwer'
3 WHERE roll_no ='001';
```

The "Run all" button is visible in the top right of the editor. Below the editor, the "History" tab is active, showing a table of executed queries:

Script	Date	Status	Runtime
Untitled - 1	Oct 31, 2022 3:21:43 PM	✓ 1	0.005 s
UPDATE users SET password='wenwer' WHERE roll_no ='001'		✓	0.005 s

☰

[🔗](#)

[SQL](#)

[🔧](#)

[🔗](#)

[📄](#)

[🔗](#)

[📄](#)

[💡](#)

Load Data

Load History

Tables

Views

Indexes

Aliases

MQTs

Sequences

Application objects

CQG39702.USERS

Back

[🗑️](#) [Export to CSV](#) [⬇️](#)

USERNAME	EMAIL	ROLL_NO	PASSWORD
arvind@gmail.com	arvind	003	wefw
hari@gmail.com	hari	001	wenwer
nehanth@gmail.com	nehanth	004	wefwe
prem@gmail.com	prem	002	erhffg

IBM Db2 on Cloud

SQL

Filter objects

CQG39702

Tables

USERS

Views

MQTs

Aliases

Nicknames

\*Untitled - 1

1DELETE FROM users

2WHERE roll\_no = '001';

Syntax assistant

Run all

History

Find history

Script	Date	Status	Runtime
^Untitled - 1	Oct 31, 2022 3:22:36 PM	✓1	0.009 s
DELETE FROM users WHERE roll_no = '001'		✓	0.009 s
^Untitled - 1	Oct 31, 2022 3:21:43 PM	✓1	0.005 s
UPDATE users SET password='wenwei' WHERE roll_no = '001'		✓	0.005 s



### Question 3. Connect python code to db2

Code:

```
from flask import Flask
import ibm_db

app = Flask(__name__)

try:
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=SSL;SSLServerCer
tificate=DigiCertGlobalRootCA.crt;UID=cqg39702;PWD=hIRRYoYSNHJxjqQq", "", "")
except:
    print("Unable to connect: ", ibm_db.conn_error())



@app.route("/")
def dashboard():
    return "Hello there!"

if __name__ == '__main__':
    app.run(debug=True)
```

#### Question 4.

Create a flask app with registration page, login page and welcome page. By default load the registration page once the user enters all the fields store the data in database and navigate to login page authenticate username and password. If the user is valid show the welcome page.

Existing database:

CQG39702.USERS				Back
				 Export to CSV 
USERNAME	EMAIL	ROLL_NO	PASSWORD	
arvind@gmail.com	arvind	003	wefw	
hari@gmail.com	hari	001	erhf	
nehanth@gmail.com	nehanth	004	wefwe	
prem@gmail.com	prem	002	erhffg	

## User.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import ibm_db

app = Flask(__name__)
app.secret_key = 'a'

try:
    conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=b0aebb68-94fa-46ec-a1fc-
1c999edb6187.c3n41cmd0nqnrk39u98g.databases.appdomain.cloud;PORT=31249;SECURITY=SSL;SSLServerCer
tificate=DigiCertGlobalRootCA.crt;UID=cqg39702;PWD=hIRRYoYSNHJxjqQq", "", "")
except:
    print("Unable to connect: ", ibm_db.conn_error())

@app.route("/")
def dash():
    return render_template('register.html', msg=" ")

@app.route("/register", methods=['GET', 'POST'])
def register():
    error = None
    if request.method == 'POST':
```

```

username = request.form['username']
email = request.form['email']
roll_number = request.form['rollnumber']
password = request.form['password']
sql = "SELECT * FROM user WHERE roll_number=?"
prep_stmt = ibm_db.prepare(conn, sql)
ibm_db.bind_param(prepare_stmt, 1, roll_number)
ibm_db.execute(prepare_stmt)
account = ibm_db.fetch_assoc(prepare_stmt)
print(account)
if account:
    error = "Account already exists! Log in to continue !"
else:
    insert_sql = "INSERT INTO user values(?,?,?,?)"
    prep_stmt = ibm_db.prepare(conn, insert_sql)
    ibm_db.bind_param(prepare_stmt, 1, email)
    ibm_db.bind_param(prepare_stmt, 2, username)
    ibm_db.bind_param(prepare_stmt, 3, roll_number)
    ibm_db.bind_param(prepare_stmt, 4, password)
    ibm_db.execute(prepare_stmt)
    flash(" Registration successfull. Log in to continue !")
return render_template('login.html', error=error)

```

```

@app.route('/login', methods=['GET', 'POST'])

```

```
def login():
    error = None
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        sql = "SELECT * FROM user WHERE username=? AND password=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, username)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
    if account:
        session['Loggedin'] = True
        session['id'] = account['USERNAME']
        session["username"] = account["USERNAME"]
        flash("Logged in successfully!")
        return redirect(url_for("welcome_page"))
    else:
        error = "Incorrect username / password"
        return render_template('login.html', error=error)

@app.route('/welcome')
def welcome_page():
```

```
    return render_template("welcome.html", user=session['id'])

if __name__ == '__main__':
    app.run(debug=True)
```

## register.html

```
<html>

<head>
    <title>Registration page</title>
    <link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
</head>

<body>
    <div class="center"> <br /><br />
        <form action="/register" method="POST">
            <h2>Student Registration Portal</h2> <br /><br />
            <p> Enter Email ID:</p>
            <input type="email" name="email" />
            <p> Enter Username:</p>
            <input type="text" name="username" />
```

```

        <p> Enter Password:</p>
        <input type="password" name="password" />
        <p> Enter Roll number:</p>
        <input type="text" name="rollnumber" /> </br></br></br>
        <input type="submit" value="submit" />
    </form>
</div>
</body>

</html>

```

login.html

```

<html>

<head>
    <title>CEG login</title>
    <link rel="stylesheet" href="{{url_for('static', filename='style.css')}}">
</head>

<body>
    {% if error %}
    <p><strong style="color:blue">Error</strong>: {{error}}</p> {% endif %}

```

```

{% with messages = get_flashed_messages() %}
{% if messages %}
{% for message in messages %}
<p style="color:yellow">{{ message }}</p> {% endfor %}
{% endif %} {% endwith %}
<div class="center"> <br /><br />
    <form action="/login" method="POST">
        <h1>Login</h1> <br /><br />
        <p> Enter Username:</p>
        <input type="text" name="username" />
        <p> Enter Password:</p>
        <input type="password" name="password" /> </br></br></br>
        <input type="submit" value="submit" />
    </form>
</div>
</body>

</html>

```

## Index.html

```

<!DOCTYPE html>
<html lang="en">

```



```
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/semantic-
ui@2.4.2/dist/semantic.min.css">
  <script src="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.js"></script>
  <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js"></script>
  <script> function delete_flash(flash) { $(flash).parent().remove() } </script>
</head>

<body>
  <div style="margin:13%;" class="ui placeholder segment">
    <div class="ui icon header">

      <i class="user icon"></i>
      Welcome back,
    </div>
    <a href="{ { url_for('logout',usernameOrEmail=usernameOrEmail) } }">
      <div class="ui blue button">Log Out</div>
    </a>
  </div>
</body>
```

```
</html>
```

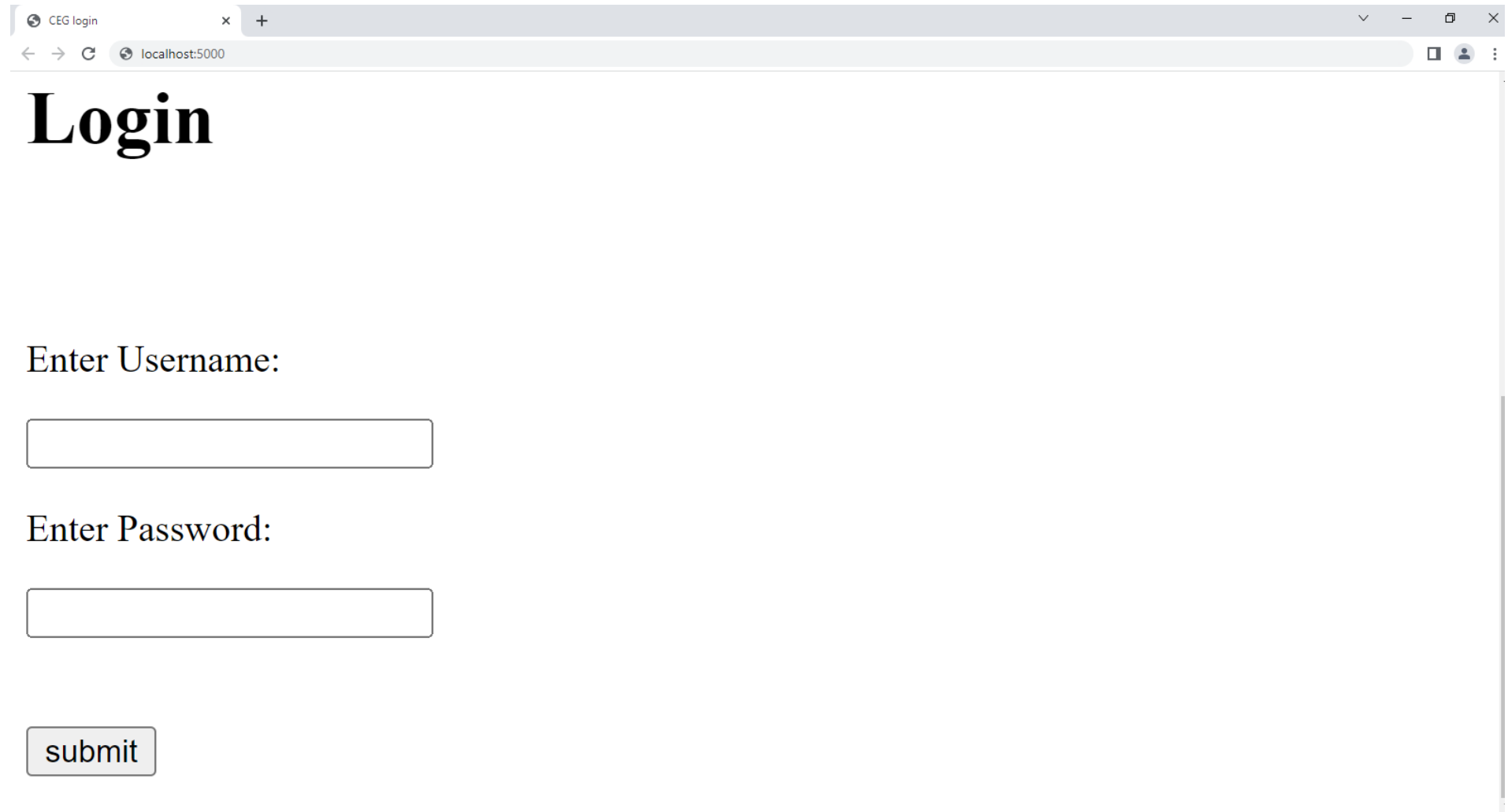
## Style.css

```
body{
    color: black;
}
.center {
    margin: auto;
    width: 15%;
}

h2 {
    text-align: center;
}

input {
    border-radius: 5px;
    padding: 7px;
}
```

## OUTPUT



A screenshot of a web browser window. The browser has a single tab titled "CEG login". The address bar shows "localhost:5000". The page content includes a large "Login" heading, followed by "Enter Username:" and an input field, then "Enter Password:" and another input field, and finally a "submit" button.

CEG login x +

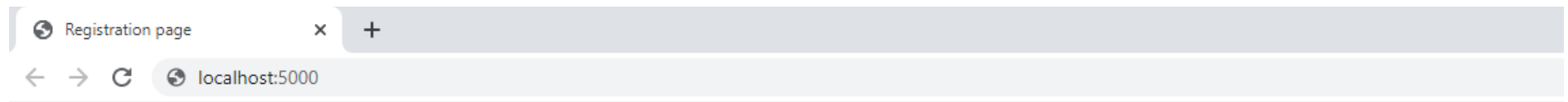
localhost:5000

# Login

Enter Username:

Enter Password:

submit



## Student Registration Portal

Enter Email ID:

Enter Username:

Enter Password:

Enter Roll number:

submit



Welcome back,

Log Out